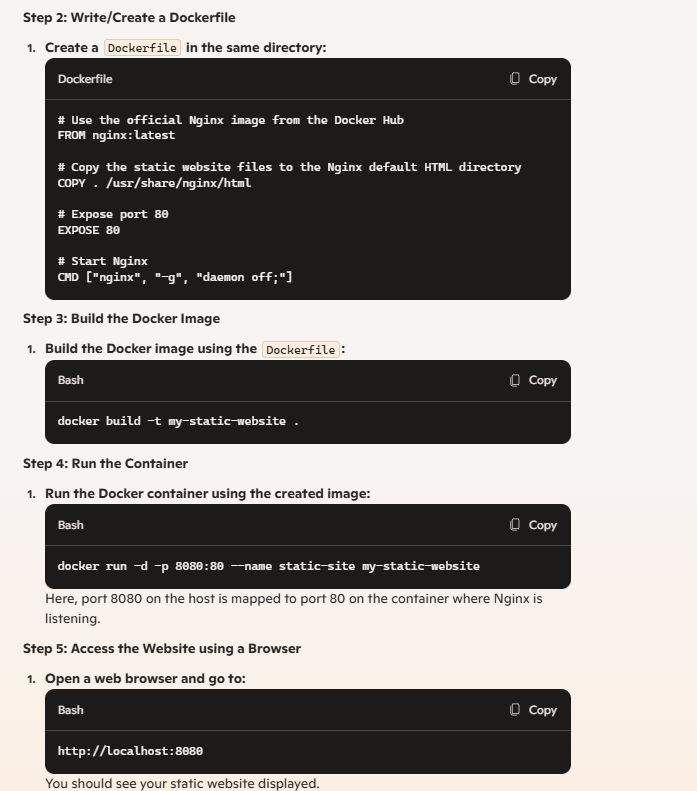
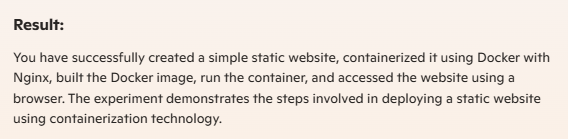
DOCKER

1. Create a Static Website and Containerize, Build & Serve it using Docker. Tasks: 1. Create a Simple Static Website (index.html file) with basic HTML content. 2. Write/create a Dockerfile to serve the website using Nginx. 3. Build the Docker Image 4. Run the container: 5. Access the Website using a Browser

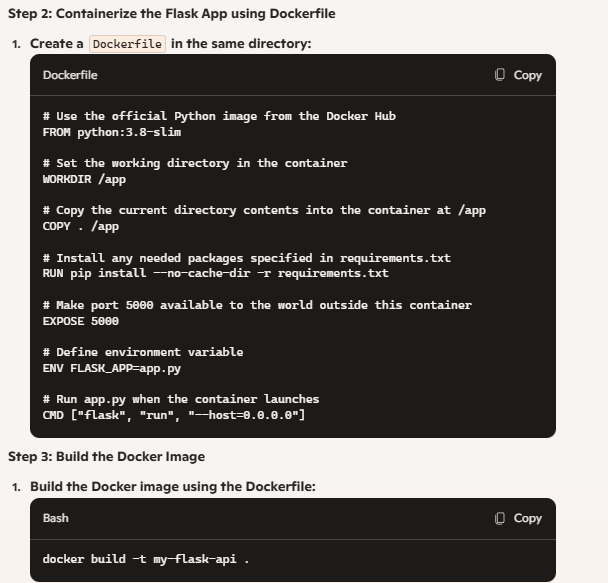


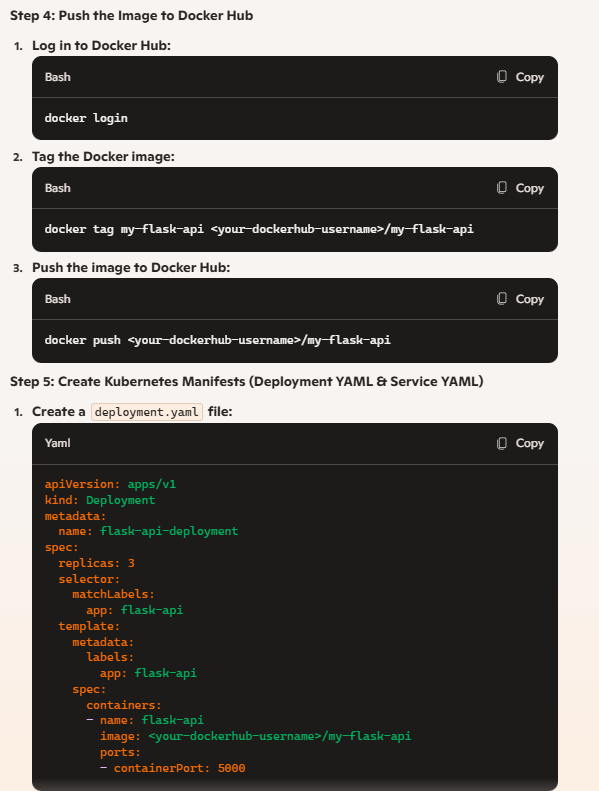




1. Create a Simple Python Flask API, Containerize the Application, Build & Push the Image using Docker and Deploy the Application using Kubernetes. Tasks: 1. Create a Simple Flask API by writing a Python file (app.py) with basic endpoints. 2. Containerize the Flask App using Dockerfile. 3. Build the Image using Docker. 4. Push the Image to Docker Hub. 5. Create Kubernetes manifests (Deployment YAML & Service YAML) to deploy the application. 6. Apply the Manifests and Access the API via NodePort.

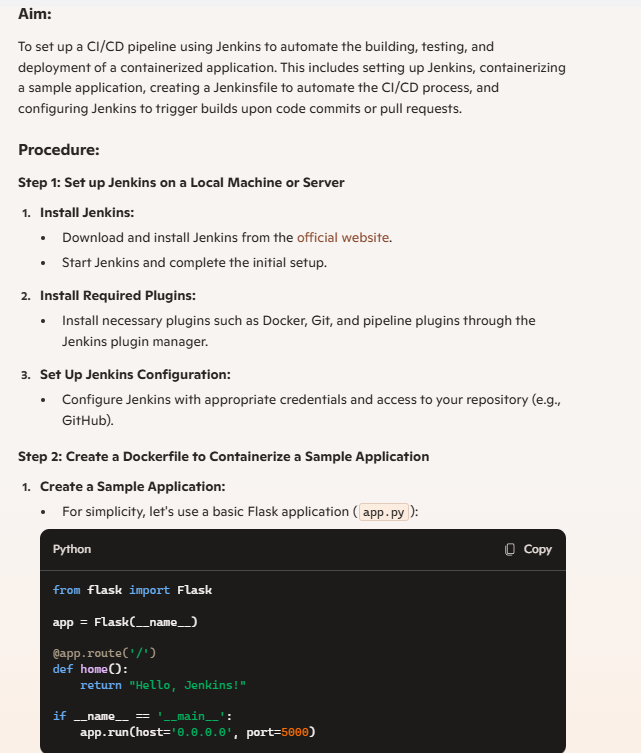


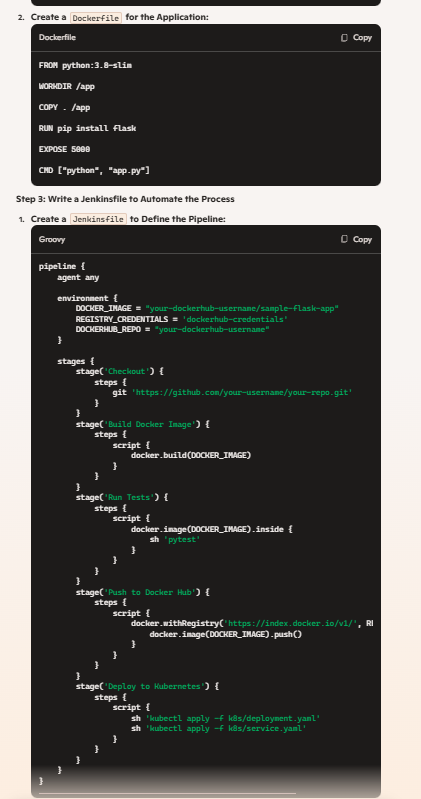


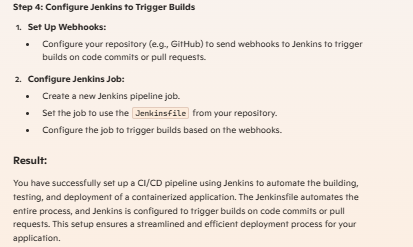




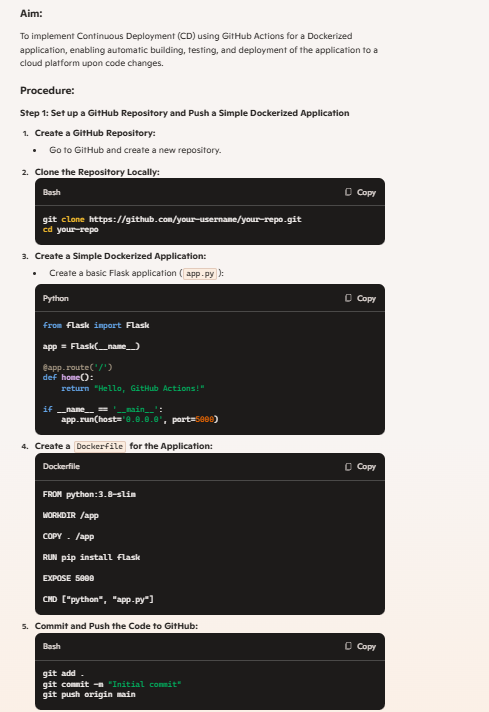
1. Set up a CI/CD pipeline to automate the building, testing, and deployment of a containerized application. Tasks: 1. Set up Jenkins on a local machine or server 2. Create a Dockerfile to containerize a sample application. 3. Write a Jenkinsfile to automate the process of building the Docker container, running tests, and deploying to a cloud platform (e.g., AWS or GCP). 4. Configure Jenkins to trigger builds upon code commits or pull requests.



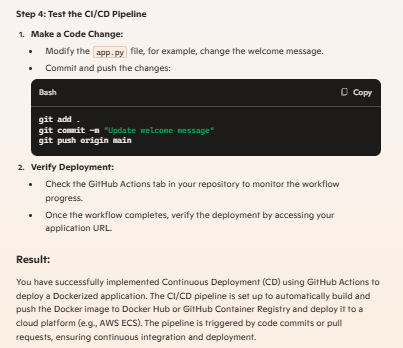




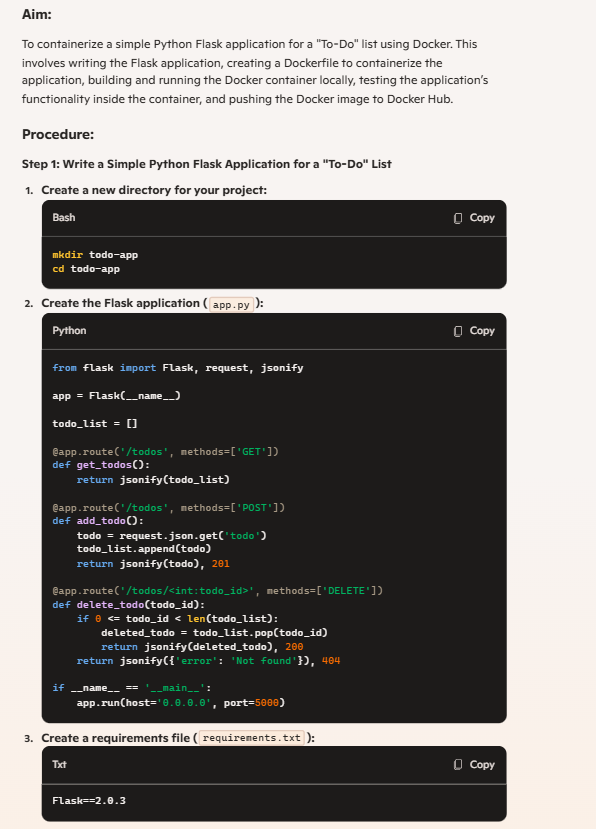
1. Implement Continuous Deployment using GitHub Actions to deploy a Dockerized application. Tasks: 1. Set up a GitHub repository and push a simple Dockerized application. 2. Create a GitHub Actions workflow to automatically build and push the Docker image to Docker Hub or GitHub Container Registry. 3. Automate deployment to a cloud platform (e.g., AWS ECS, Azure Kubernetes Service, or Google Kubernetes Engine). 4. Test the CI/CD pipeline by pushing new code changes and verifying that the deployment occurs automatically

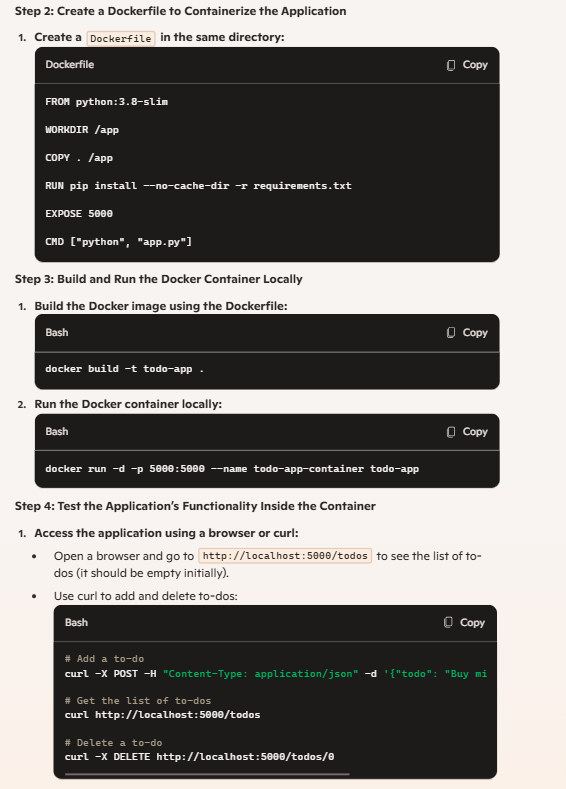
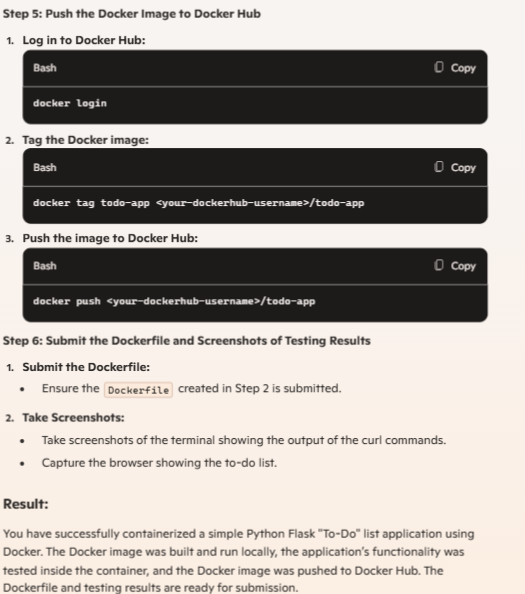






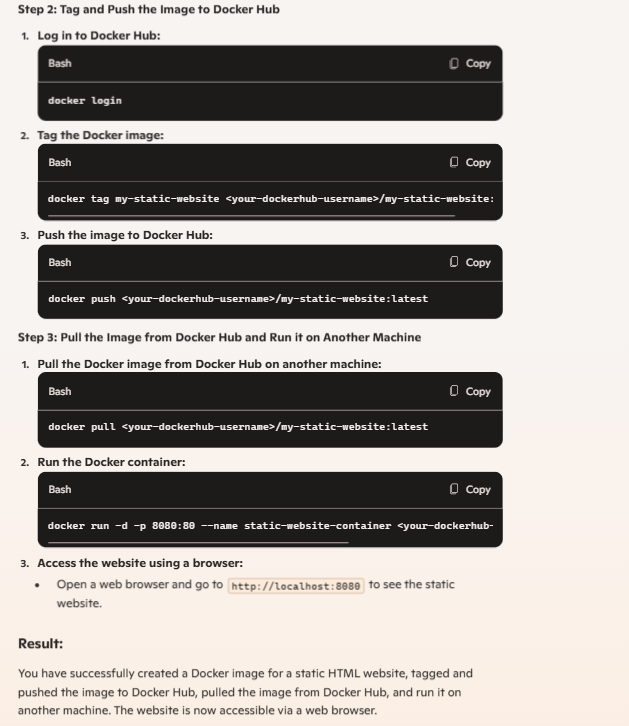
1. Containerize a Python Flask Application Using Docker Tasks: • Write a simple Python Flask application for a "To-Do" list. • Create a Dockerfile to containerize the application. • Build and run the Docker container locally. • Test the application’s functionality inside the container. • Push the Docker image to Docker Hub. • Submit the Dockerfile and screenshots of testing results

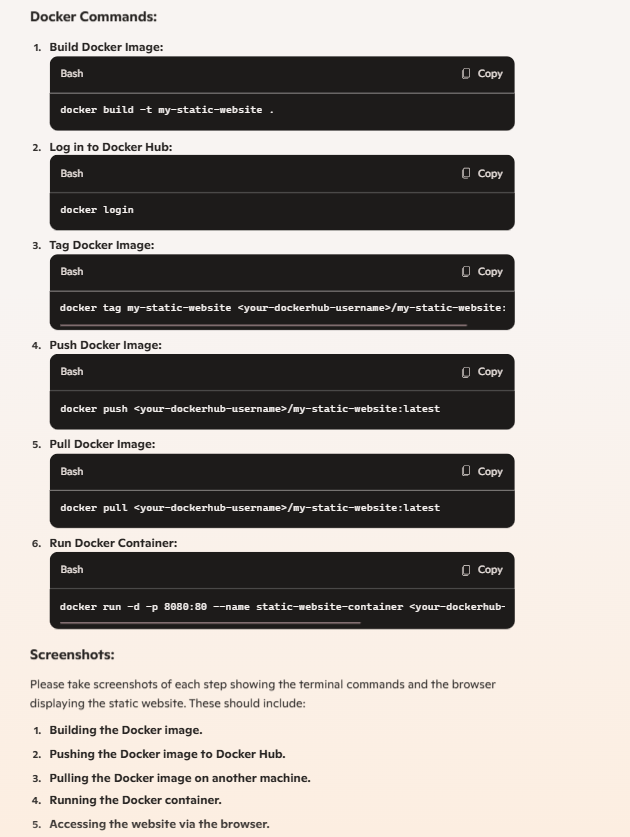


1. Push and Pull Docker Images Using Docker Hub Tasks: • Create a Docker image for a static HTML website. • Tag and push the image to Docker Hub. • Pull the image from Docker Hub and run it on another machine. • Submit Docker commands and screenshots for each step.







1. Deploy a Multi-Container Application Using Kubernetes Tasks: • Create a multi-container application with a frontend, backend, Docker, Kubernetes CO3 and database. • Define a Kubernetes YAML file for deployment and services. • Deploy the application using kubectl commands. • Monitor the pods and services status. • Scale the frontend container to handle increased load. • Submit YAML files and screenshots of the Kubernetes dashboard.

